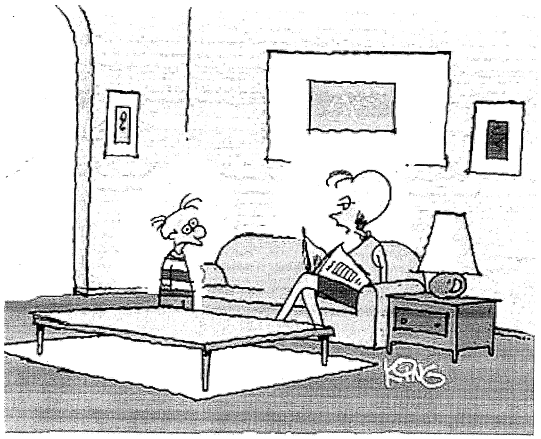
Never forget ....



"No, you weren't downloaded.
Your were born."

Parallel Computing vs Distributed Computing: a Great Confusion?

Michel RAYNAL

raynal@irisa.fr

Institut Universitaire de France

IRISA, Université de Rennes, France

Hong Kong Polytechnic University (Poly U)

Content

- This is a *position paper*
- The target is teaching (not writing applications)
- What is parallel computing?
- What is distributed computing?
- They are different topics!
- A curriculum for distributed computing
- Do not forget: algorithms are the essence of computing
- Conclusion

Two quotes

- *Only that which has no History can be defined.*
  Friedrich Nietzsche ((1884–1900)
- *Every sentence I utter must be understood not as an affirmation, but as a question.*
  Niels Bohr (1885-1962)

Two analogies/metaphors

- Application = write a text
  - ⋆ Orthography
  - ⋆ Grammar
- Application = cook a main plate
  - ⋆ Fish/meat
  - ⋆ Vegetables

Part I

Parallel computing

From the Web

"Introduction to parallel computing"
by Blaise Barney, Lawrence Livermore National Lab

https://computing.llnl.gov/tutorials/parallel_comp/

"This tutorial is the first of eight tutorials in the 4+ day "Using LLNL's Supercomputers" workshop. It is intended to provide only a very quick overview of the extensive and broad topic of Parallel Computing"

A view from the Web (cont'd)

- Serial computing
- Parallel computer
- Motivations
  - ⋆ Save time/money
  - ⋆ Solve larger problems
  - ⋆ Take advantage on multiplicity resources
  - ⋆ Better use of hardware
  - ⋆ etc.

Parallel computing

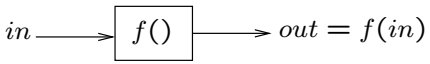
Given a problem

- Solve it with a seq computer would be very inefficient
- Hence: divide the problem in sub-problems and solve as many as possible of them in parallel
- This is a (very sensible) DESIGN CHOICE
- The pb could have been solved by a seq computer (assuming we have enough time!)

Parallel computing is about mastering efficiency

From sequential to parallel computing

The atom of a computation is the notion of a function



A function $f()$ (sequential computing)

- Parallel computing is on concepts, methods, and tools addressing the implementation of $f()$ when one can use several processors
- Remark: the input x is inherently centralized its scattering on processors is a design choice
- Simple example: parallel sorting

In parallel computing

- Inputs are inherently centralized (their possible distribution is a design choice)
- Outputs are a "function" of the inputs
- Key concepts:
  - ⋆ Vector processing, systolic programming
  - ⋆ Load balancing, scheduling
  - ⋆ task graph
  - ⋆ etc.

Part II

Distributed computing

On distributed computing (1)

- Distributed computing arises when one has to solve a problem in terms of distributed entities (usually called processors, nodes, processes, actors, agents, sensors, peers, etc.) such that each entity has only a partial knowledge of the many parameters involved in the problem that has to be solved

On distributed computing (2)

- In any distributed computing problem, there are several computing entities, and each of them has to locally take a decision, whose scope is global
- DC motto: "Think globally, act locally"
- The uncertainty is not under the control of the programmer, it is created by the geographical scattering of the computing entities, the asynchrony of their communication, their mobility, the fact that each entity has only its own inputs, etc.

Distributed computing is about mastering uncertainty

Two fundational papers

- Lamport L., Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558-565, 1978

  Partial order, scalar clocks, state machine duplication

- Fischer M.J., Lynch N.A., and Paterson M.S., Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374-382, 1985

  Notion of bivalence

  Both won the Dijkstra Prize (#1 and #2)

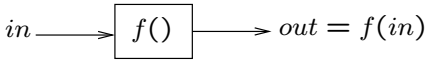
On distributed computing (cont'd)

- The world is distributed (not centralized!)
- More and more applications are distributed
- The inputs are distributed (e.g., sensors)
- What is the basic unit of a distributed computation?

Returning to the basics

In sequential computing

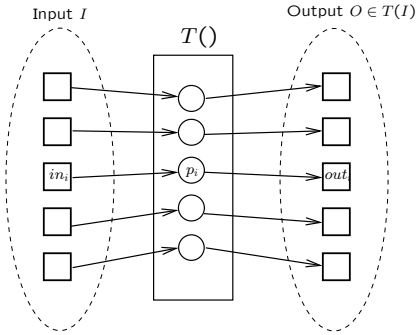
The atom of a computation is the notion of a function



A function $f()$ (sequential computing)

From seq computing to distributed computing

The concept of a distributed task



A task $T()$ (distributed computing)

$$[out_1, ..., out_n] = T([in_1, ..., in_n])$$

A simple example: consensus

- Very basic agreement problem
- Each process is assumed to propose a value
- The processes have to agree on the same value, which has to be a proposed value

$$[out_1, ..., out_n] = T([in_1, ..., in_n])$$

where $\exists v \in \{in_1, ..., in_n\} : \ out_1 = \cdots = out_n = v$
(intrinsic non-determinism)

- Failure-free systems: trivial
- Crash-prone asynchronous systems: impossible (FLP'85)

In distributed computing

- Inputs are always distributed
- Outputs depend on the inputs and the environment
- A few important concepts:
  - ⋆ Notion of a local computation
  - ⋆ Notion of failure (crash, Byzantine behavior)
  - ⋆ Symmetry breaking
  - ⋆ Agreement
  - ⋆ etc.

Part III

Teaching Distributed Computing

Returning to the basics

- Sequential computing $\rightarrow$ sequential algorithms
  - ⋆ What can be done, and what cannot be done (e.g., sorting pb)
  - ⋆ Success stories:
    - ∗ complexity theory
    - ∗ Chomsky's hierarchy (FSA $\rightarrow$ Turing machines)
    - ∗ Church-Turing's thesis (Notion of universality)
- Distributed computing: teach basic algorithms!

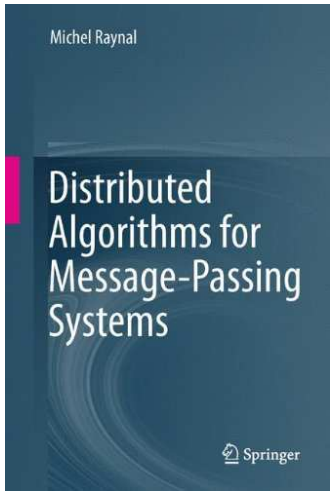
Undergraduate level: basic notions

- Distributed graph algorithms $\rightarrow$ locality notion
- Quick look at "possible/impossible" (e.g., election)
- Nature of distributed computing:
  - ⋆ Notion of causal dependence
  - ⋆ Notion of a global state
- Two particular problems:
  - ⋆ Distributed mutual exclusion
  - ⋆ Termination detection

A curriculum: Message-passing, asynchrony, no failure

- Distributed Graph algorithms
- Logical times and global states in distributed systems
- Mutual exclusion and resource allocation
- High-level communication abstractions
- Detection of properties of distributed executions
- Distributed shared memory (consistency conditions)

Undergraduate level: a personal textbook



Distributed Algorithms for Message-Passing Systems

by Michel Raynal

Springer, 513 pages, 2013

ISBN: 978-3-642-38122-5-2

Graduate level: failure-prone systems

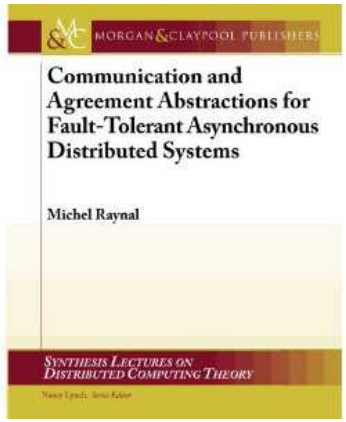
- When communication is through a shared memory
- When communication is through message-passing

A curriculum: message-passing and failures

- The register abstraction
- The uniform reliable broadcast abstraction
- Agreement abstractions

Keywords: agreement abstraction, anonymous systems, asynchronous systems, atomicity, causal message delivery order, communication failures, consensus, failure detectors, process failures, randomized algorithms, reliable broadcast, total message delivery order.

Graduate level: Message-passing and failures



Communication and Agreement Abstractions for Fault-Tolerant Asynchronous Distributed Systems

by Michel Raynal

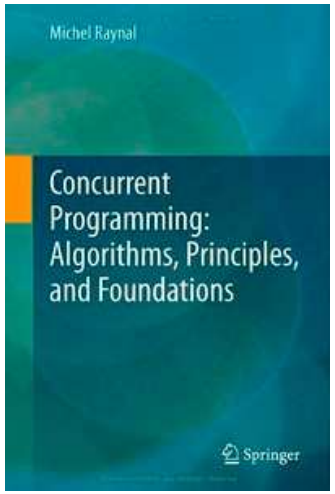
Morgan & Clapypool, 251 pages, 2010

ISBN: 978-1-60845293-4

A curriculum: shared memory and failures

- Lock-based synchronization
- On the foundations side: the atomicity concept
- Mutex-free synchronization, and associated progress conditions (obstruction-freedom, lock-freedom, and wait-freedom)
- The transactional memory approach
- On the foundations side: from safe bits to atomic registers
- On the foundations side: the computability power of concurrent objects

Graduate level: Concurrent progamming



Concurrent Programming: Algorithms, Principles and Foundations

by Michel Raynal

Springer, 531 pages, 2013

ISBN: 978-3-642-32026-2

Books of friends and colleagues

- Attiya H. and Welch J.L., *Distributed computing: fundamentals, simulations and advanced topics, (2nd Edition)*, Wiley-Interscience, 414 pages, 2004
- Cachin, C., Guerraoui R., and Rodrigues L., *Introduction to reliable and secure distributed programming*, Springer, 367 pages, 2012
- Herlihy M. and Shavit N., *The art of multiprocessor programming*. Morgan Kaufmann, 508 pages, 2008
- Kshemkalyani A.D. and Singhal M., *Distributed computing: principles, algorithms and systems*. Cambridge University Press, 736 pages, 2008
- Lynch N. A., *Distributed Algorithms*. Morgan Kaufmann Pub., San Francisco (CA), 872 pages, 1996
- Santoro N., *Design and analysis of distributed algorithms*. Wiley, 589 pages, 2007
- Taubenfeld G., *Synchronization algorithms and concurrent programming*. Pearson Education/Prentice Hall, 423 pages, 2006

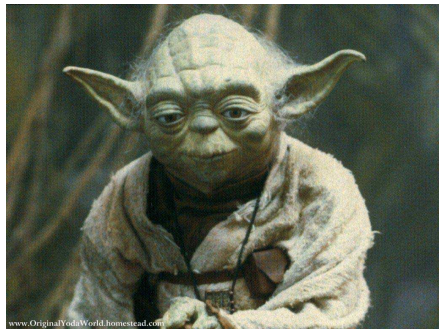
Part IV

Conclusion

Conclusion

- This was a teaching-oriented position
- Do not confuse applications and basic concepts
- Why students have to know technology
- Why students have to assimilate concepts
- Teaching is not an accumulation of facts

More important, HE told me



"In the long run, ...
maybe concepts are the most important"

and then, HE added

"Prediction is difficult, ...
mainly when it concerns the future!"